An Architecture Pattern Enabling Safety at Lower Cost and with Higher Performance

Rolf Johansson, Jörg Kaiser, António Casimiro, Renato Librino, Kenneth Östberg, José Rufino, and Pedro Costa

***Abstract*— In both avionic and automotive systems, it might become very costly and/or restricting the functional performance, to prove functions safe in all operational conditions and for 100% of the mission time. This is especially true if the quality of sensor data and of communication data may vary very much. One way to solve this trade-off paradox is to leave part of the safety assessment from design-time to run-time. This paper proposes a general architectural pattern for this, and also how to instantiate this pattern in Integrated Modular Avionics (IMA) for the avionic domain, and in AUTOSAR for the automotive domain. The solutions imply some extensions of ARINC 653 and of AUTOSAR respectively, but they are not in conflict with the existing concepts. The proposed solutions are also fully in-line what is prescribed by the standards for functional safety of the two domains.**

***Index Terms*—Safety Integrity, IMA, AUTOSAR,**

I. Introduction

In both the industry fields of Avionics and of Automotive, there has been established a norm of having integrated, rather than federated, Electrical/Electronic (E/E) architectures. In the avionic field this is called Integrated Modular Avionics (IMA) [6] and is today often following the ARINC 653 specifications [7], and in the automotive field it is the AUTOSAR specifications constituting the state-of-practice.

One advantage with an integrated architecture is that it is possible to increase the number of vehicle functions and still decrease the number of computing nodes, often called LRUs (Line Replaceable Units) or ECUs (Electronic Control Units). Different functions may be realized by architectural elements (sensors, actuators, computing components, communication components, etc.) that may be shared among several functions. However, in the transition from a federated to an integrated pattern, the way to ensure functional safety has become more complex. Instead of directly transfer the safety arguing responsibility to a node supplier, the integrated pattern calls for a component-based approach in both the design itself and in the safety case generation. Furthermore, achieving functional safety is a goal that often is in conflict with high performance and low cost.

In this paper we outline an architectural pattern that is possible to apply for both IMA and AUTOSAR, and that helps to resolve the paradox of getting functional safety together with low cost and high functional performance.

II. Arguing Safety

The problem how to prove that a solution is functionally safe, is defined specifically by applicable standards in each domain. For road vehicles, functional safety is defined by ISO 26262 [10], and the instruction how to apply component-based safety arguing is so far limited to what is stated as Safety Element out of Context (SEooC) in the informative part 10 of the standard. For the avionic domain there are several applicable standards. The IMA perspective is found in RTCA/D0-297: “Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations”. This standard refers to other standards such as SAE International's Aerospace Recommended Practice (ARP) 4754 on “Certification Considerations for Highly-Integrated or Complex Aircraft Systems”, and SAE ARP 4761 on “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment”. These in turn refer to RTCA/DO-178: “Software Considerations in Airborne Systems and Equipment Certification”.

On a high level, one can say that arguing safety is similar in the two domains. They both rely on a Hazard Analysis & and Risk Assessment (HARA) of the vehicle functions, resulting in required risk reduction by means of safety integrity levels for the realizing architectural components. In the automotive domain the safety integrity levels are called Automotive Safety Integrity Level (ASIL) [10], and in the avionic domain they are called Development Assurance Level (DAL) [12]. The default alternative is to use a fault avoidance argumentation based on the fact that all used components, each conform to the required safety integrity level. This can be combined with a fault-tolerance argumentation based on introducing redundancy and hence lower the required integrity level of the components. To integrate components having different integrity levels on the same platform, freedom of

This work was supported by the project KARYON (reference no.: 288195), funded by Seventh Framework Program (FP7/2007-2013) of the European Commission.

Rolf Johansson and Kenneth Östberg are with SP Technical Research Institute of Sweden (e-mail: rolf.johansson@sp.se, kenneth.ostberg@sp.se).

Jörg Kaiser is with Otto von Guericke Universität Magdeburg (email: kaiser@ivs.cs.uni-magdeburg.de)

António Casimiro and José Rufino are with Faculdade de Ciências da Univesidade de Lisboa (email: casim@di.fc.ul.pt, ruf@di.fc.ul.pt)

Renato Librino is with 4SGroup (email: renato.librino@4sgroup.it)

Pedro Costa is with GMV Skysoft (email: pedro.costa@gmv.com)

interference has to be shown. This is a major concern in RTCA/DO-297.

III. Determine the Required Risk Reduction (Safety Integrity Level safety)

In both domains, the determination of safety integrity level is based on a severity classification of the possible failures of the vehicle function under consideration. In the avionic domain this directly leads to one of the Development Assurance Levels in the interval between A (required by a catastrophic failure) and E (when there is no safety effect required). In the automotive domain the severity classification of the vehicle function has to be evaluated together with assumptions on how often such a failure is critical among the driving scenarios, and with an assumption how well the driver can compensate for the failure. These three factors together, lead to an Automotive Safety Integrity Level in the range between ASIL D (highest requirement on risk reduction) and QM (no requirement on risk reduction).

In both domains, a high requirement on risk reduction implies a high safety integrity level requirement (formulated by DAL or by ASIL) on the used components. It is needed to show that the vehicle and its components fulfill all requirements on safety integrity levels. This assessment is completely done in design-time, and shows that the vehicle always is functional safe with respect to the complete scope of the functions considered in the Hazard Analysis.

IV. Extending the architectural pattern

Building vehicles where the complete scope of all functions are proven safe in all operational conditions and for 100% of the mission time, might become very costly and/or restricting the functional performance. One way to solve this trade-off paradox is to leave part of the safety assessment from design-time to run-time. By letting the architecture itself in run-time measure the provided safety integrity levels from the components, it can enable adjustment of the different functions such that their required safety integrity levels are met. Let each vehicle function have a number of predefined levels of performance (levels of service), for which the resulting hazard analyses are different. If we then in run-time can measure for which levels of service/performance the safety integrity requirements are fulfilled, the vehicle can be proven safe once we can guarantee that all functions always are forced to a level of service/performance that is considered safe. What we need for this is an architectural pattern enabling the complete measuring of the safety integrity levels of the architectural elements, and a way to ensure that all functions operates on a level of service/performance that can be proven safe.

This paper shows the general architectural pattern, and also outlines how it can be instantiated into the state-of-practice in the integrated architecture of the domains of automotive (AUTOSAR) and of avionics (IMA realized by ARINC 653), respectively.

A. Design-Time vs. Run-Time

Even if the strategy proposed in this paper is that there is a run-time check of what safety integrity levels that are met in order to match with the appropriate performance level, all possible results have to be assessed in design-time. In both the avionic and the automotive domain, a complete functional assessment has to be done in design-time according to the respective reference life-cycles. The solution presented in this paper is fully aligned with this. When we say that we leave part of the safety assessment to run-time, this implies that all the possible alternatives for performance levels and also the mechanisms to determine the right levels, all are assessed in design time.

B. Relation to functional safety standards of today

In both DO178 and in ISO26262, there is a concept of integrity levels (DAL / ASIL), to allocate requirements on the reference life cycle in order to argue sufficiently absence of systematic design faults. When we suggest having several levels of performance implying different required safety integrity levels on the output of some components, this implies that the DAL/ASIL applicable for the design of each component will be the highest safety integrity level among the possible ones. The concept of adjusting the levels of performance to the run-time available safety integrity levels is hence not primarily applicable to handle systematic design faults, but to take care of the varying quality of data due to the varying amount of redundancy sources and of varying quality of sensors and communication links.

Especially from the software design point of view, we assume that what is prescribed as needed to argue for the highest applicable safety integrity level, still must be implemented as well in the application components as in the platform software (ARINC653/AUTOSAR). This requirement of design according to the highest safety integrity level is of course applicable to the redundancy mechanisms checking the quality of data, and to the platform mechanisms determining the appropriate level of service/performance. This is further elaborated in the section below: A general pattern.

The consequence of having a number of different levels of performance is that the Hazard Analysis and Risk Assessment has to be done completely for each considered level. The stages prescribed in ARP4754 or ISO26262 part 3 thus have to be done not only for the functions, but for each level of performance for each function. Furthermore, each transition between two levels of performance has to be considered, as elaborated below in the section: Scalability and timing.

Regarding the safety standards applicable today in the domains of avionic and automotive, we conclude that they are fully in-line with both the concept of several levels of performance, and with the architectural pattern we outline in this paper.

C. Relation to existing patterns for mixed criticality

Mixed criticality [5] is the concept of allowing applications with different levels of criticality (safety integrity) to co-exist on the same system. In both ARINC 653 and in AUTOSAR there are mechanisms to deal with this problem. The assumed problem to deal with is to guarantee freedom of interference between application components designed to different levels of safety integrity. The solutions in both ARINC 653 and in AUTOSAR are to provide mechanisms for handling time and space partitioning.

However, when introducing several levels of performance, each implying different requirements on safety integrity levels of the output of some components, this is not the classical problem of mixed criticality. Even if we have a mixture of different levels of safety integrity among the components co-existing on the same platform, their requirements on the safety integrity of the absence of systematic design faults are all on the same level (the highest among alternatives). Hence we do not need to prove freedom of interference between components designed according to different DAL or ASIL, because that is not the case. All components used for a certain function are designed according to the safety integrity level that is applicable for the highest level of performance.

The mechanisms for enabling time and space partitioning are of course still important when arguing safety, even in our proposed pattern, but they are not affected by the introduction of several levels of performance with different safety integrity level requirements on some data signals.

D. Scalability and timing

Given the fundamental idea of leaving part of the safety assessment for the run-time, raises potential issues of scalability and timing.

Scalability issues stem from the fact that some system resources will be required for performing the run-time safety assessment. For instance, it will be necessary to collect measurements of available integrity levels, and it will be necessary to store information (defined in design time) concerning the safety integrity requirements for each level of service. This has essentially practical implications (availability of enough memory and computing power), which could limit the applicability of the approach. Fortunately, the effective requirements grow linearly with the number of components for which integrity has to be assessed, which is also limited by the available resources. In fact, the additional resources required for assessing safety in run-time are necessarily a very small fraction of the resources required by the components itself.

Timing issues are more important in this context because it is necessary to argue about functional safety, for which they have to be considered. As mentioned before, in design time it is necessary to assess, for each function, that it will perform safe in each possible performance level. But this is not enough. Given that, for each function, there will be run-time changes of the performance level, the analysis must take into account the time that it takes to complete these changes. For instance, if in run-time it is detected that some component is not performing with the required integrity level, then it will be necessary to change the performance level of all the functions that are affected by this integrity degradation, and this has to be done within some limited amount of time. Otherwise, the functions would continue to perform in some inadequate level for an uncertain amount of time, clearly outside the safety analysis performed in design time.

The general pattern must hence provide the means to address these timeliness requirements. In addition, it is necessary to discuss the implications of the (bounded) amount of time that is necessary to change the performance level of some function. A sufficient condition for ensuring timely detection of changes in the safety integrity levels, and consequent timely change of performance level, is that it is possible to perform a timing analysis of all the involved system components, deriving upper execution bounds. In particular, this includes the component responsible for performing the safety assessment, which is always involved in the process. If some function has to be performed with some minimal performance level, then it must also be possible to perform such timing analysis for all the implied components. The time that will be necessary to switch among different performance levels will typically be close the execution periods of the system components. In comparison with the typical latency of physical processes (such as braking, deviating from an obstacle, etc), these periods are much smaller. This means that the safety analysis will be, for the relevant part, still valid. In any case, the time that it takes to perform a change in the performance level can also be considered in the design of the functions, so that this is accounted in safety margins.

V. A General Pattern

Each vehicle function is realized by a set of interconnected sensors, actuators and software components. The software components have an interface making them possible to be allocated on any platform node. The outputs of every sensor, and of every software component, are duplets consisting of both the nominal output and of an attribute stating the estimation of the corresponding safety integrity level. All redundancy mechanisms in the architecture such as: sensor fusion, voting, consistency checking, etc., are evaluating the consistency of the nominal values and calculates a resulting determination of the safety integrity level value. All safety integrity level values of a computing node are checked by a safety manager that is part of the platform specification. The safety manager compares in run-time that each provided safety integrity level of every output value, is high enough for the current scope of vehicle functions. If some of the provided safety integrity levels are too low, the safety manager tells the application mode managers to change to a level where the respective functions are considered safe. If all provided safety integrity levels are high enough for a higher level of service/performance than the actual for some functions, the safety manager tells the application mode managers to change accordingly.

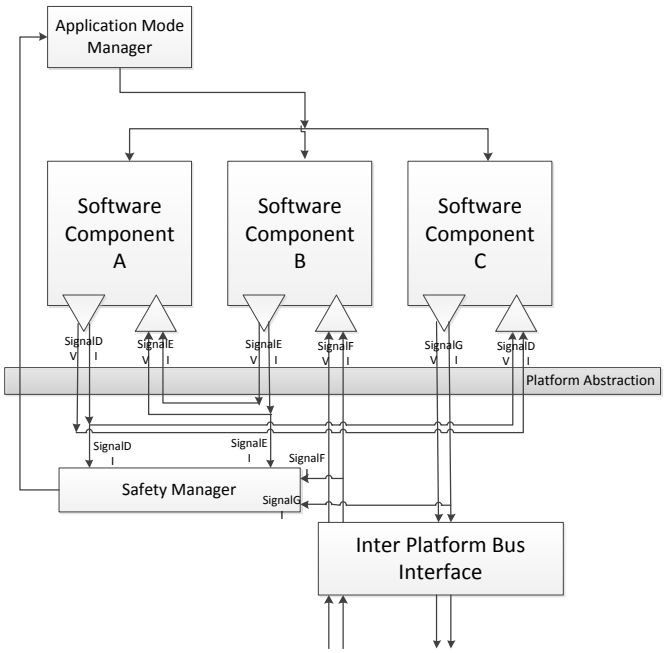


Fig. 1. The figure of a general architecture pattern. Each software component communicates via ports with signals. Each signal is a duplet with a nominal value (V) and a safety integrity level attribute (I). The safety manager checks all safety integrity level values, and decides what function modes that are safe. The mode manager tells each software component about the mode decisions

A. Derivation of the Safety Integrity Attributes

As argued before, the extension of safety assessment from design-time to run-time allows for relaxing overly restrictive assumptions about the required integrity status of the overall complex control systems. However, it requires the continuous monitoring and evaluation of the integrity attribute during run-time. Fig.1 shows that for every signal generated by a component, there is a complementary output that provides an integrity level attribute. Roughly, this attribute represents a measure for the integrity of the respective signal produced by a component. When checking against allocated safety requirements, this estimation needs to be done against the defined discrete levels of the applicable standard. For both the standards we have five levels to consider. However, when deriving the safety integrity attributes we may use more fine-grained estimations. In this paper we call such a more fine-grained value, the Validity of a signal.

The assumed structure of a complex component comprises the acquisition and computational components from a sensor that captures a real-world entity to the component that outputs an application relevant data element (signal). The architectural element outputs the nominal signal (V) together with an integrity attribute (I) that enables the safety assessment at run time. To generate the integrity attribute, the component needs a self-assessment mechanism. Further, a nominal value affected by a failure may pass a filter to mitigate or mask the effect. We therefore distinguish between a detection mechanism and a filter mechanism. The checking mechanism detects a failure without affecting the respective signal. It modifies the integrity attribute only. The filter is a general abstraction of a component that mitigates or eliminates the effect of a failure. Typically, detectors and filters are integrated in a fault-tolerance mechanism. We separate the aspects of awareness and treatment because these are different concerns and the separation allows for more freedom of design. e.g. omitting a filter completely in the component for handling the failure in a subsequent stage. Fig. 2 depicts this general structure of such an element.

The run-time assessment mechanisms are based on the specification and quantification of design-time assumptions. During design-time an engineer has to answer questions like "which failure types are affecting the components and what is their impact?", "How are these failures detected and how good the detection mechanisms need to be?" and " How is the data conditioned and filtered to compensate the effects of failures?". Based on these assumptions the engineer adjusts the quality of the component's outgoing data at run-time to the integrity requirements. In our approach, these engineering assumptions are quantified to allow a comprehensible assessment. Assumptions are quantified in a failure model, a quantification of the detection capabilities and the filter characteristics. This is particularly needed when such a component is used in a larger setting or will be reused in another design. As an example we examine a typical component where the input data is provided by a sensor.

We distinguish two flows of information in Fig.3. The lower part generates the nominal data output while the upper part is devoted to the calculation of the integrity attribute. In this paper we will focus on the definition and transformation of the parameters defining the integrity attribute.

We are considering a data centric failure model [2] which specifies failures in terms of how they affect the data e.g. according to an anticipated signal behaviour. The starting point is the identification of relevant sensor failures. Sensors deliver continuous values that may be affected by e.g. noise, offsets, spikes and outliers. A detailed discussion about sensor failure modes is given in [3]. An assessment vector quantifies for each failure type a number that characterizes the anticipated severity and the occurrence probability of this failure. This is comparable to a risk priority number in the FMEA (Failure Mode and Effects Analysis) scheme [4]. Different from FMEA, we provide a scheme that allows combining multiple failure types and using this for run time assessment, while FMEA maps multiple failure types to the static worst case risk priority number.

For describing the detector and filter characteristics we provide transformation matrices that modify the assessment vector for the respective stages. In case of a detector, the matrix specifies the ratio of correctly detected failures versus the wrongly and not detected failures. This statically sets the upper and lower bounds for the integrity attribute and modifies the assessment vector accordingly. The filter matrix defines the impact that the filter may have on the signal, i.e. the degree of suppressing a faulty signal. A filter that operates as a failure masking mechanism will raise the lower bound for the validity because it eliminates faulty values. Applied to the assessment vector it modifies the respective elements related to the affected failure types. The assessment vector finally holds for each failure type an element that describes which effect this specific failure will have on the final integrity attribute. It should be noted that this number includes the capabilities of the detector and filter with respect to the particular failure. The integration stage collapses the vector representation to a single scalar integrity attribute. This stage uses a selection vector that holds weights for each failure type and therefore allows a further restriction to relevant failure types. E.g. for long term navigation, single outliers of a localization sensor may not be as relevant as a constant offset failure. However, outliers may have a high impact on the validity because of their amplitude. Thus, an outlier would decrease the integrity attribute to an inacceptable low level although it would not be relevant. Another application may need a high validity of differential positions. Here, constant offsets would not play a major role. The selection vector can adjust these different application needs. In the end, we obtain what we call the system validity, a measure of how good the detection and filter mechanisms will deal with failures.

So far, all the information that is captured in the assessment vector, the transformation matrices and the selection vector is available at design time and quantifies of the engineering assumptions about relevant failure types, their impact, the quality of the detection mechanism and the power of the filter in suppressing the effect of failures.

In the conventional approach, the outcome of this analysis would be compared to a required integrity and in the case of a match, the design would be accepted. This implies that the assumptions about the failures, the detection and filter capabilities are worst-case assumptions and require a substantial amount of resources to keep the bounds. If the design does not fulfill the static worst-case requirements, the design needs to be changed. This may require substantial redundant resources or more expensive components even though the actual operation would not even come close to the worst-case bounds in by far the most cases. In the proposed architectural pattern, the numerical representation of the integrity is exploited to detect at run time whether the component will meet the integrity requirements. If the dynamically derived integrity attribute falls below a certain threshold, we are able provide the countermeasures on a higher level, i.e. on the level of provided services that may need to be degraded (see also the example in the next section B).

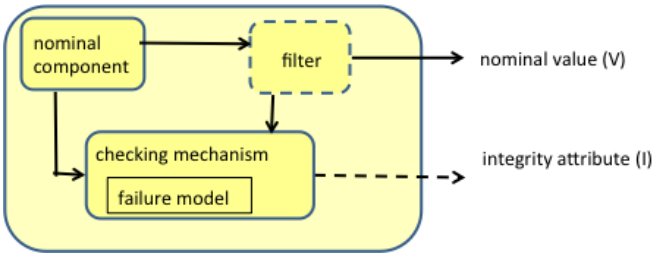


Fig. 2. Estimating the signal quality by a checking mechanism.

At run time, the detector will provide a result for each failure that it is able to detect. These outcomes are stored in a vector with the same dimension as the assessment vector. The elements of the assessment vector are applied as weights to this vector to form the validity vector. The validity vector represents the actual estimated validity as a result of the detector stage. It is transferred to the filter stage where a similar calculation is performed. The final validity vector holds a dynamic estimate about the validity of the generated nominal data item with respect to each failure type. Finally, this vector is converted into a single scalar number which represents the integrity attribute. The details of the assignment of values and the operations defined by the failure algebra are beyond the scope of this paper. They are provided in [5].

B. Relating the validity estimation to the Integrity Attributes

In a safety-critical system, the hazard analysis will result in a set of functional failures that should not occur. In the coming phases these restrictions are broken down to what failures should not occur for the elements in the chosen architecture. Still, all these failures are restricted by the safety integrity level attributes telling how sure one have to be that these failures will not occur. The question is how sure we can be at run-time about the absence of these failures, if we cannot guarantee at design-time that this will always hold. We now will show, how the dynamic validity can be exploited to quantify these requirements. Let us first use a simple data centric model of failure. Failures may be specified in terms of its amplitude, e.g. the error exceeds ±5% of the true value or in duration e.g. that they will not last for more than 10ms. According to the goal of our architectural pattern, we enable the dynamic change of the Level of Service (LoS) to maintain

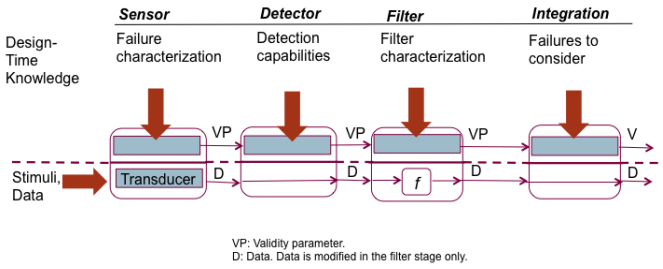
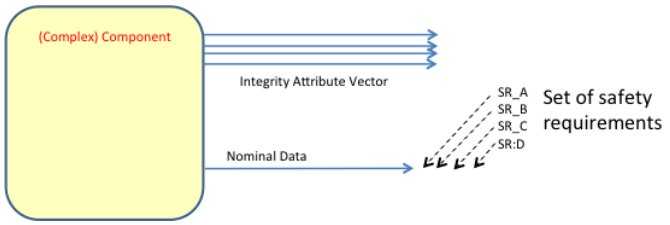


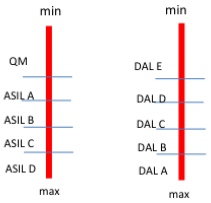
Fig. 3. Computational chain for a sensor processing component.

a required level of safety. For the information generated by a component that contributes to a function we need specifying

different levels of confidence that the data value is within certain bounds. Fig. 4 illustrates this.



a.) A complex component has to meet several safety requirements



b.) Example of defining 4 thresholds for different safety integrity levels in the automotive and avionics terminology

Fig. 4 Example of a complex component used in applications running in multiple integrity levels

The complex component provides a nominal output that is subject of multiple safety requirements ranging from the highest ASIL D to QM. Integrity attributes are provided at separate outputs. They are represented as an integrity attribute vector that holds for each failure the confidence that it does not exceed the predefined bounds. Considering the thresholds in Fig. 4b, we model failure classes that e.g. do not deviate more than ±2%, ±5%, ±10%, ±20% and ±30% from the true value for ASIL D, C, B, A, and QM respectively. The integrity attribute vector holds the respective confidence values that these values are indeed within the specified bounds. If the confidence drops below a certain value in the most demanding safety class, this confidence value may well be within the bounds of a lower one. This would be reflected by the integrity attribute of the respective failure class. In this case, that the integrity attribute is too low, it may be necessary to switch to another level of service.

The notion of validity that was introduced before can be mapped to this more safety centric perspective. Given that the requirements of the safety integrity level concerning the error of the nominal value are specified, the validity calculus allows defining the failure characteristics through the computational chain. The failure model will be adapted to the thresholds of the amplitudes. Consider that the failure may not exceed ±2% of the true value and we want to be sure about this at a certain confidence level. This affects the choice of the sensor, the detector and the filter. The validity calculus allows to quantify these effects. At the sensor, an assessment vector represents the sensor characteristics for each failure type in terms of amplitude and of occurrence probability. If the requirements specify a very low margin on the amplitude of failure, noise, small offsets and drift have to be considered. We have to consult the failure model to see which failures are effective in such tight bounds. This will be analysed statically for each deviation addressed in the example above. In the end, we will have the design-time assurance that at run-time, under all anticipated conditions, the output of the component will have the integrity level indicated in the respective integrity attribute vector. If the safety requirements are very high, as it is e.g. required for ASIL D, the quality of the sensor and the mechanisms to mask a failure to always be within the bounds must be very high. I.e. we may need massive redundancy of expensive components. The problem is, that even a less expensive sensor will deliver results within tight bounds most of the time with sufficient probability. Detection and filters will detect or suppress failures sufficiently most of the time with sufficient probability. Thus the probability of a failure at run time will meet the ASIL D requirements most of the time but violations may happen. Conventional systems usually do not use the mechanisms to assess the integrity level at run-time. Instead they guarantee by design time analysis that such violations will never happen.

In contrast, our system explores the safety-cost-performance trade-off by offering multiple levels of service. Without sacrificing safety, we may run a function with a lower integrity attribute at a lower level of service. Prerequisite for this is the run-time assessment. The run-time integrity attribute output will give us, for each of the bounds specified in the example above, the confidence that the actual nominal value really is within the respective bounds. Thus, the design-time analysis is the basis to derive the run time values. The quality of detection and filter mechanisms are included in this assessment. A detector, for example, is characterized by the probability to deliver false {positives, negatives} and true {positives, negatives}. It is clear that for guaranteeing a very high confidence in the checked data, these values must be adequate. An ideal detector with no false positives and negatives will detect each failure correctly and therefore will transform the integrity attribute of a nominal value to "0" in case of a failure detected and "1" if no failure is detected. Real detectors, of course, will have weaker bounds. For a filter we similarly specify the ability to mask a failure. Details of these calculations are presented in [5]. As a consequence of the quantitative representation of engineering assumptions and the tight interplay between design time assurance and run time assessment, we can adjust the needed confidence according to the needs of the safety requirements. If such a dynamic value, represented in the integrity attribute, is not sufficient for a function running in a certain assurance level we have to switch to a lower level of service.

VI. Instantiation in AUTOSAR

AUTOSAR is a de facto standard in the automotive domain enabling an integrated architecture. It defines how to specify application software components (SWC) that are reallocatable. All software component inputs and outputs are as data elements through ports. Our suggested pattern implies that each data element should consist of a duplet: the existing

nominal data element, complemented by an ASIL attribute. It is the responsibility of each SWC to compute the corresponding ASIL values. If there is no redundancy implemented, this means that the input ASIL values are inherited for the output values. Otherwise, ASIL decomposition is applied according to the algebra as defined in the ISO26262 standard. We furthermore propose a new basic software module (BSW): Safety Manager. This is in line with existing managers among the BSWs today. This new safety manager will compare the computed ASIL value of each signal with the required ones, that are stored for each function level of performance/service. In AUTOSAR today, all data elements are already connected to the run-time environment (RTE), constituting the interface for the BSW. Our pattern hence implies that the RTE will be extended with the requirement to connect all ASIL values to the Safety Manager.

The concept of modes is on three hierarchical levels in AUTOSAR: Vehicle Modes, Application Modes and BSW Modes, see Fig. 5. It's only the BSW modes that are standardized in the AUTOSAR set of specifications. It is assumed that different applications have implemented modes in their definitions. The control of these modes is within the application implementation, i.e. it is implemented by application software components (not in the platform). On an even higher level than application modes, there is Vehicles modes, which are global for the entire vehicle. As seen in the Fig. 5 there might be influences between modes on all these levels.

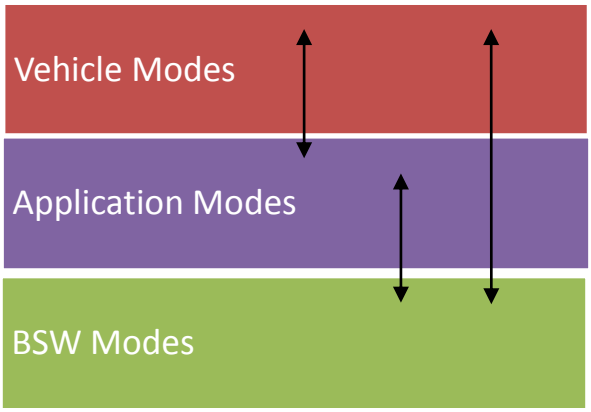


Fig. 5 How the different kinds of modes in AUTOSAR may influence each other.

We assume that the existing concept of application modes can be used for forcing all applicable SWCs to a mode corresponding to the level of service/performance as considered the highest and still safe by the Safety Manager. This is one extra connection between a SWC and the RTE, but it is in-line with the existing pattern and hence considered as an attractive extension of AUTOSAR.

As we don't consider checking spatial and temporal interference between SWCs, the normal BSW mode managers as specified by the AUTOSAR standards such as: ECU State Manager, BSW Mode Manager, Communication Manager and Watchdog Manager are not of importance here. The application mode managers are implemented as ordinary software components and communicate with other software components via RTE.

VII. Instantiation in IMA

Currently, in the avionic domain IMA is considered to be implemented by means of the ARINC 653 specification [7], which determines that applications are functionally separated in logical containers, called partitions. One goal of partitioning is to ensure the containment of faults in the domain in which they occur. Partitioning in logical containers implies non-interference of applications' execution in the time domain and the usage of separated memory and input/output addressing spaces [8].

Application software components are hosted in partitions. All software components inputs and outputs are as data elements, using the inter-partition communication services provided by the ARINC 653 Application Executive (APEX) application programming interface primitives. In particular, we propose to take advantage of the *sampling ports* services to extend the software component port values with a DAL attribute, as illustrated in Fig. 6.

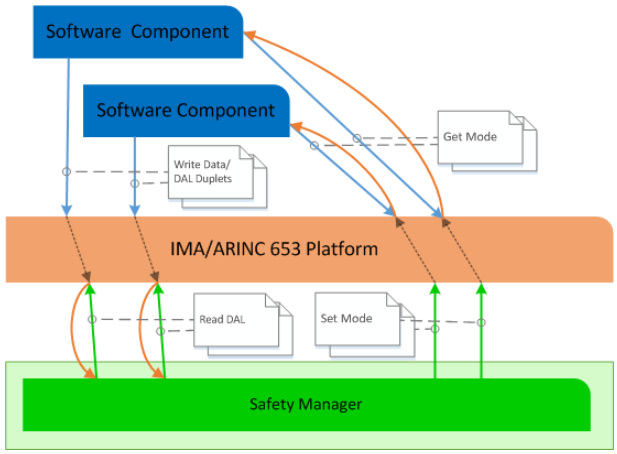


Fig. 6 Instantiation in IMA/ARINC 653 platforms

The Safety Manager will also be hosted in a partition and it will monitor the DAL attributes of all software component signals [9]. This can be easily achieved by platform configuration, allowing inter-partition communication services to deliver the components' nominal output and DAL duplets both to the destination components and to the Safety Manager, which for will read the DAL attribute, as illustrated in Fig 6.

A similar approach can be followed when the Safety Manager needs to change the performance/service level and set a new application mode. This action, which may also take advantage of inter-partition communication services, is also illustrated in Fig. 6. The Safety Manager sets the highest possible safe level of service/performance and the components change to the corresponding mode.

With this approach the instantiation of the general architecture pattern in IMA/ARINC 653 platforms does not require any modification or extension of the platform itself; only a platform configuration action is needed.

VIII. Conclusion

This paper summarizes how to apply a new architectural pattern as an extension to existing state-of-practice in the domains of avionics (IMA) and of automotive (AUTOSAR). The pattern is applicable to solve the problem when it is hard to show in design-time that a high safety integrity is met under all circumstances and for 100% of the mission. By introducing different levels of service/performance each having different implications on needed safety integrity, high performance most of the time can be combined with guaranteed functional safety all of the time.

The pattern constitutes that every signal value that is the candidate for different requirements on safety integrity levels, should be evaluated by a redundancy mechanism capable to calculate a run-time estimation of the actual provided safety integrity level.

Furthermore, the pattern implies the introduction of a Safety manager taking care of checking the safety integrity level attribute of every system signal. The safety manager compares during run-time all currently provided safety integrity levels with those derived from the break-down of the hazard analyses of the different levels of performance/service of the vehicle functions. As a result of the comparison, the safety manager enforces all functions to operate in the highest performance/service level that is still safe, by means of application mode managers.

References

[1] J. Barhorst, T. Belote, P. Binns, P. Hoffman, J. Paunicka, P. Sarathy, J.S.P. Stanfill, J.S.P. Stuart, and R. Urzi, "A research agenda for mixed-criticality systems (2009)", http://www.cse.wustl.edu/~cdgill/CPSWEEK09_MCAR/RBO-09-130%20Joint%20MCAR%20White%20Paper%20PA%20approved.pdf, white paper.

[2] K. Ni, N.Ramanathan, M. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen und M. Srivastava, "Sensor network data fault types" in *ACM Transactions on Sensor Networks (TOSN*), Bd. 5, Nr. 3, pp. 1-29, 2009.

[3] Zug, Sebastian; Dietrich, André; Kaiser, Jörg, "Fault-Handling in Networked Sensor Systems" (book chapter), Rigatos, Gerasimos (Hrsg.): Fault Diagnosis in Robotic and Industrial Systems, Concept Press Ltd., St. Franklin, Australia, 2012.

[4] D. H. Stamatis, "Failure Mode and Effect Analysis: FMEA from Theory to Execution", 2 Hrsg., {ASQ} Quality Press, 2003.

[5] Brade, Tino; Zug, Sebastian; Kaiser, Jörg: "Validity-based failure algebra for distributed sensor systems" in Proc. of the IEEE 32st Symposium on Reliable Distributed Systems, Braga, Portugal, 2013.

[6] AEEC: Design guidance for Integrated Modular Avionics. ARINC Report 651-1 (November 1997)

[7] AEEC, Avionics application software standard interface, part 1 – required services, ARINC Spec. 653P1-2 (Mar. 2006).

[8] J. Rufino, J. Craveiro, P. Verissimo, Architecting robustness and timeliness in a new generation of aerospace systems, in: A. Casimiro, R. de Lemos, C. Gacek (Eds.), Architecting Dependable Systems VII, Vol. 6420 of LNCS, Springer, 2010, pp. 146–170. doi:10.1007/978-3-642-17245-8_7.

[9] P. Nóbrega da Costa, J. P. Craveiro, A. Casimiro, and J. Rufino, "Safety Kernel for Cooperative Sensor-Based Systems," in *Safecomp 2013 Workshop on Architecting Safety in Collaborative Mobile Systems (ASCoMS)*, Toulouse, France, Sep. 2013.

[10] ISO: Road vehicles — Functional safety — International Standard ISO 26262, part 1-9 (Nov. 2011).

[11] The AUTOSAR development partnership: www.autosar.org

[12] SAE: Guidelines for Development of Civil Aircraft and Systems, SAE ARP 4754A (Dec. 2010).